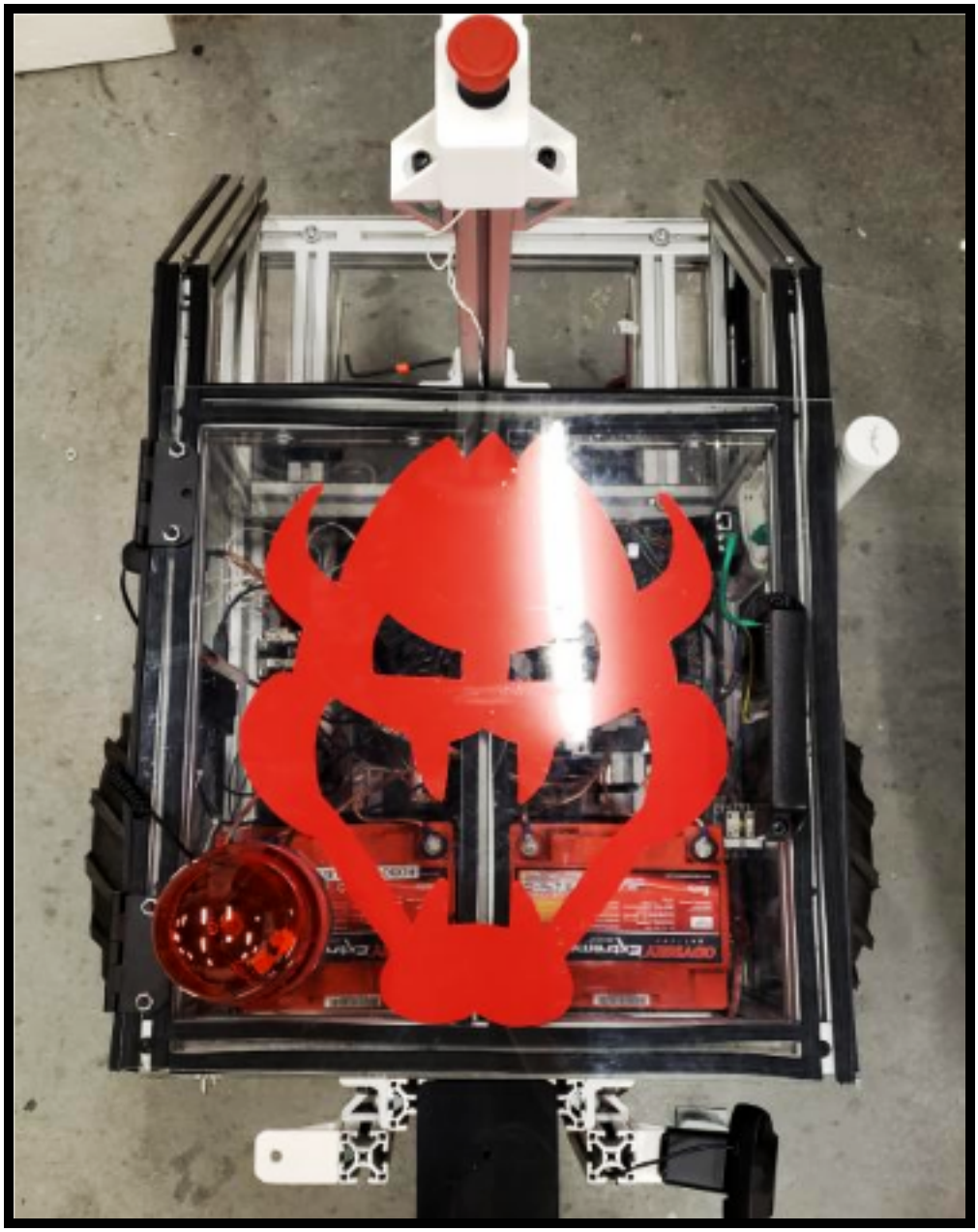
BOWSER

Jade Zsiros robotics@ucf.edu	Captain
Brian Blalock	VC, Computer Vision
Darien Craig	VC, Electronics, Sensors
Don Vo	Software, sensors, SLAM
Scott Bell	Computer vision
Salomon Hassidoff	Mechanical Engineering
Tim Lin	Software, sensors, SLAM
Alan Mark	Mechanical Engineering
Tan Thu	Documentation
Amran Subramanian	Software



I certify that the work done by all students on this project is consistent with a senior design course and that the vehicle has been significantly modified for this year's competition.

Crystal Maraj
cmaraj@ist.ucf.edu

Introduction

At the beginning of the 2018 - 2019 school year, the Robotics Club at UCF founded its first prototype team - a group of students brought together by the desire to complete a wide variety of small projects quickly, with each unit being limited to three weeks. However, between school, sporadically gaining members, and life causing some students to miss meetings at inconvenient times, it became apparent that the prototype team was not convenient to a college setting. With only one semester left in the school year, and no one with a mechanical background, the team decided to attempt a larger challenge by refurbishing and reprogramming a legacy IGVC platform, Metaknight.

For the past six months, we have rebuilt, replaced, recoded, and recruited in order to try to complete our platform in half the permitted time. With a fresh influx of programmers and mechanical engineers, we worked to select new parts and build a code base that was simple and elegant. In the end, we had replaced enough of the robot that we decided to name it something new. For this year's 2019 IGVC competition, we are competing with Bowser.

Team Organization

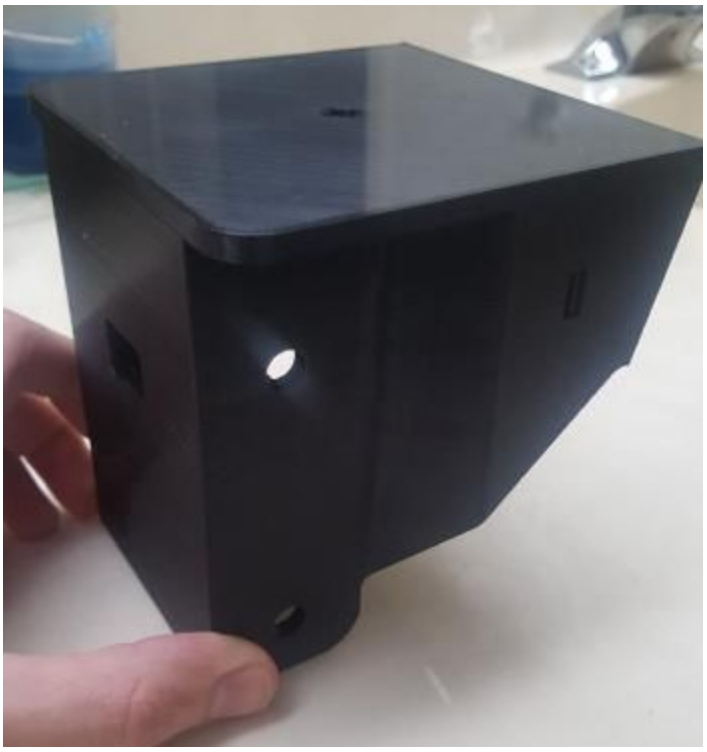
The IGVC team at the Robotics Club at the University of Central Florida reformed in January of this year, starting with a bare metal frame. Our design process started with sensors, figuring what we would need to code, wire, and mount, so we could design and build all of them simultaneously. The team was then split into mechanical, electrical, and software teams, and designed our components as detailed below.

Mechanical Design

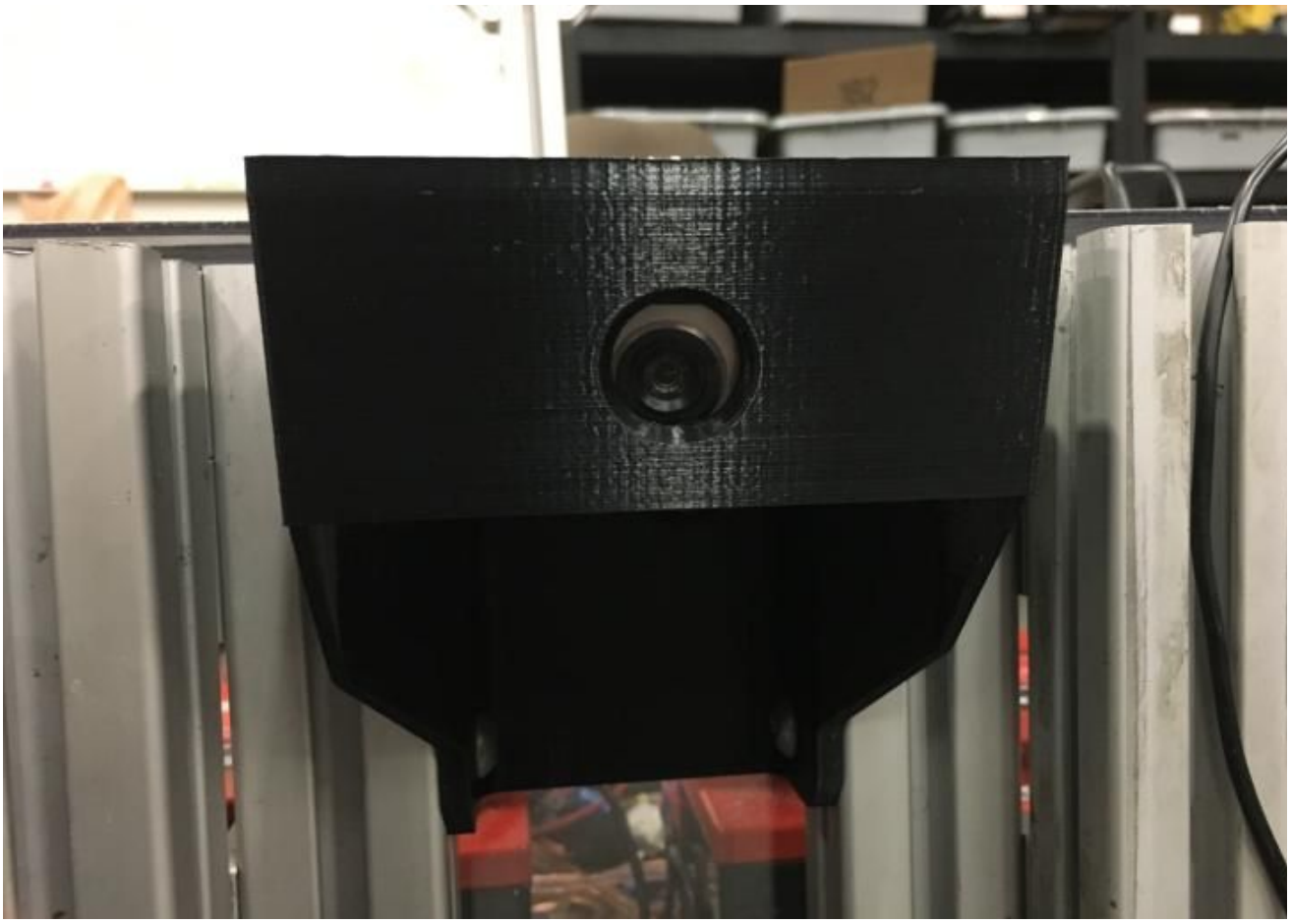
Bowser is, essentially, a large box driven by two large tiles and stabilized by two rear shopping cart wheels. Inside its weatherproof box is storage for the batteries and computing machinery, accessible by a large polycarbonate door. The frame of the robot is constructed of aluminum, and the panels are all made from polycarbonate. Where sensors are mounted, they are supported by 3D printed mounts of current student CAD design. The back of the robot, which once was a crevasse accessible for interface with the robot computer, has been turned into a shelf for the payload. Thick, 9" tires are used to ensure we are able to traverse not only rough terrain, but mild inclines. On the front side of the robot, a reinforced aluminum structure covered in thick rubber protrudes far enough to protect our sensors in case of a bump. A single stalk rises from inside the robot's "box", making accessible our E-Stop.

The box design was original to the previous iteration of Bowser, MetaKnight, but when replacing all of the internals, we redesigned the placement of the devices inside. We also reconsidered the placement of the sensors on the exterior of the robot, and prioritized simplicity of mount manufacture and safety of location. We decided we could alter the software to accommodate the sensors, but repairing the robot would be difficult outside of our own laboratory.

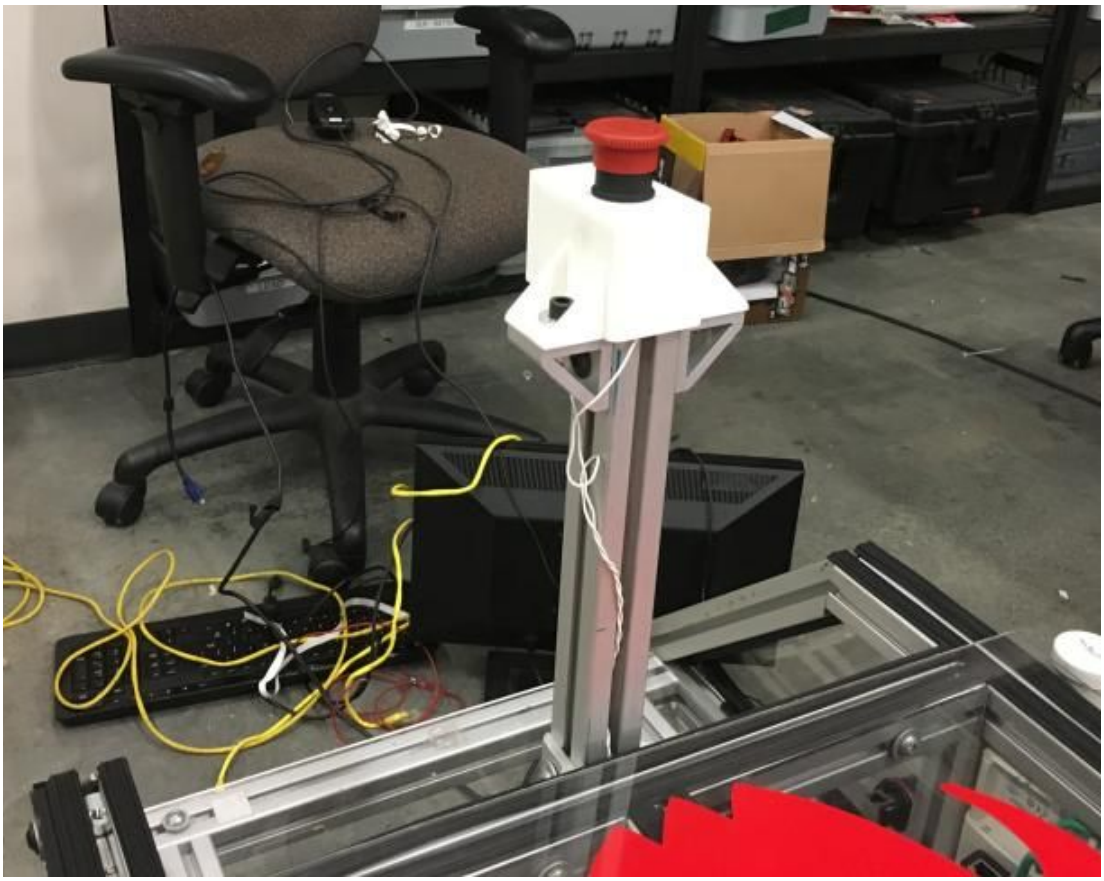
We 3D printed a multitude of sensor mounts and brackets to facilitate the changes to the robot.



The mount for the LiDAR



The LiDAR mount, which houses the front-facing camera.



The GPS/IMU mount, with E-stop on top

A top view of our three camera mounts

Software

From a software perspective, our general strategy began with completely throwing out all previous code that had been on Metaknight. None of our members had been on that team, and all of our electrical and computing hardware was new. We decided to come up with a strategy for dealing with the code from scratch. Customarily, our club uses ROS (Robot Operationg System) for most of our projects, so we decided to use that as a starting point. From there, we came up with two potential angles to approach the challenge: one based on utilizing SLAM (Simultaneous Localization And Mapping), and one based on vector field navigation. After discussing what implementing each would mean, we settled on using SLAM. We chose this option because there are existing tools in ROS to give us a leg up on starting the code, and because we felt it would be easiest to integrate our various individual code projects into the larger whole with SLAM.

While there are a multitude of ways to allow the robot to move with various combinations of sensors and software strategies, we settled on what we felt was simplest: LiDAR to detect physical obstacles, and three cameras to detect the lines on the ground. We have a camera to each side of the robot to detect our distance

from each line, and a camera to the front of the robot to detect pot holes. We combine the information from each of the four into a map. We also utilize a GPS for navigation.

As we began coding, each member of our team chose an aspect of the code to work on.

Gaussian Blur

A 9x9 Gaussian Filter, or Gaussian Blur is added to the raw video image. This filter reduces noise seen by the cameras. For example, individual blades of grass will be seen as multiple parallel lines. A Gaussian Filter blurs the image by traversing the pixels of the image and calculating the Gaussian distribution to build a convolution matrix. The matrix is then applied to the original image where each new pixel is the weighted average of neighboring pixels in its matrix, with the pixel at the center. The result is a blurred image that minimizes hard transitions and accentuates colors. We used this technique in our code for detecting various visual obstacles.

Line Detection

1. Gaussian Blur

   To reduce random noise as well as turn the distinct grass blades into a more uniform surface we apply a 9x9 gaussian filter over the image, reducing our number of false data points for later processes. Additionally, the process accentuates colors and can remove green spots from grass blades that have moved since the grass was painted.

2. White Thresholding

   As all painted obstacles are a consistent white, we run the blurred images through a threshold. This ensures that more

Pothole Detection

1. Gaussian Blur

   We blur images for pothole detection for the same reasons detailed above: to accentuate colors and remove small noise such as unpainted grass blades.

2. White threshold

   Unnecessary noise is masked by thresholding for high values in HSV color space. This reduces the chance for other noise in the grass to register with the hough circle filter.

3. Perspective modification

   Hough circle has trouble when the object you are trying to detect is flattened in any direction, as circles flat on the ground appear to our low-lying camera. To correct for this, we change the perspective by keystoning the top of the image inward, correcting circles on the floor back to circles in the image.

4. Hough Circle

   The final major step is the circle hough transform.

Obstacle Detection (cones)

1. SLAM gmapping

   A local map is created in realtime to detect cones and other high obstacles. SLAM gmapping requires two topics from ROS to function properly, laser scan data and odometry data.

2. LiDAR

   The Velodyne VLP-16 Puck is our LiDAR which provides the laser scan data. The Puck builds this map with its 360° field of view and the robot is tracked within the map using odometry data built using Bowser's wheel encoders. The range of the Puck, according to its data sheet, is roughly 100 meters. We are currently using a 15 meter range to build the map.

3. Encoders

   The wheel encoders offers data on the revolutions per minute of the wheels. Combining this information with the arclength of the wheel offers velocity data which can be used to determine how far the robot moves within the map.

   Using the difference in velocity of both wheel encoders allows us to determine information on the yaw rotation of Bowser. The calculated yaw can then be used for the twisting component of odometry, showing how the robot turns left and right. Additionally, the encoders allow us to software-limit the speed of the robot.

Electrical Design

Our electrical system runs on a consistent 12V. We have two 42 amp hour batteries, a hold over that has remained consistent for three consecutive robots. We use a Roboteq MDC2460 as our primary motor controller because of its inbuilt ability to run microbasic. This allows us to remotely control the platform without running through the main Jetson, as well as a low level software estop that can cut off all other control. We take advantage of this feature for our robot's main e-stop system.

Sensors

Onboard we have five main sensor groups:

1. Vision: we use three logitech C920 webcams. One forward facing; used primarily for pothole detection, and two side facing cameras, fed to our line detection software.
2. Lidar: Our LiDAR is a Velodyne VLP-16, which gives us a 100m horizontal range with 30º vertical range; and a ~0.3º horizontal resolution and 2.0º vertical resolution.
3. Encoders: Our encoders are separate and handled by the motor controllers with their own closed loop speed control. The motor controller also reports encoder data back to the odometry suite.

4. Inertial Measurement Unit: We use an Advanced Navigation Spatial which gives us GPS and dead reckoning as part of the same odometry package.

Failure Modes

In the case of radio silence from a sensor, our robot will stop moving until it reestablishes connection or is powered down. The robot is software-limited in speed. For other issues, we have a remote E-Stop.

Performance to Date

At the time of writing, our robot is still very much under construction. It is capable of mapping and driving.

Hardships and Issues

Many IGVC teams compete repeatedly, taking a “ship of Theseus” approach to the competition and utilizing past platforms and software to improve year after year. With only half the allotted time to start our code base from scratch, we found ourselves thoroughly investigating our various possible strategies at the beginning of the semester with the knowledge that we would have only one shot to build something worthwhile. Additionally, many of our members were new to the club, new to the school, or had never programmed seriously before. It was a learning experience for all of us.

We encountered a variety of challenges that seemed to pose a serious threat to our ability to compete. In the beginning, we had no one who had ever used anything like an IMU, and it took us weeks to be able to get it to communicate with ROS in any capacity. Additionally, we had no computer that could be used for the robot, and our school was generous enough to purchase for us an Nvidia Jetson. However, due to various factors that slowed down the process, we were finally able to begin work on the actual machine in early March. Progress was slow, and eventually, the school year came to a close. At this time, we lost the majority of our core membership to well-earned but inconvenient out-of-state internships and jobs.

In the end, we sat down and discussed seriously whether we should compete in IGVC this year. We decided that while this year we may not be a “ship of Theseus” team that can build on our prior knowledge and existing robots to achieve the highest ranks, we can only hope to begin down that road by doing our best, and our best starts with attending this year’s IGVC with open minds and an eagerness to gather what information we can to improve in the future. We are bringing some of our youngest students, even those not directly involved in the Bowser project, so that they can learn from this experience and carry it forward. We strive to become that team.